관계형 DB 설계

관계형 데이터 베이스

- 데이터 별 테이블을 생성, 테이블 사이의 종속성을 ***"관계"***로 표현하는 것이 특징인 데이터 베이스의 일종이다.
- 모든 데이터를 한 테이블에 몰아서 저장하는 방식이 아닌, 데이터를 유형 별로 분류한 뒤 각 테이블 사이의 관계를 형성시켜 데이터를 관리하는 방식이다.
- EX) MySQL, MS-SQL, MariaDB, Oracle

키(Key)

- 각 열(튜플)들을 식별할 때 사용하는 속성 값들에 해당. → 쉽게 말해 데이터 구분할 때 사용하는 값들이라 보면 됨
- 데이터는 중복된 값을 가질 수 없음. 무조건 어느 하나의 속성은 다른 속성들과 달라야 하며, 이처럼 구분할 수 있는 특성을 갖는 역할을 하는 것이 ***키(Key)***이다.

도서번호	도서이름	출판사	가격
1	축구의 역사	굿스포츠	7000
2	축구 아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
5	피겨 교본	굿스포츠	8000
6	피겨 교본, 피겨 기초	굿스포츠	8000

동일한 투플이 중복되면 안 됨

속성의 값은 단일 값이어야 함

- 위의 예시에서는 ***피겨 교본***이 중복되는 것을 확인할 수 있다. 이처럼 중복된 값을 데이터 베이스에선 허용하지 않는다.

주문

고객번호	도서번호	판매가격	주문일자
1	1	7000	2014-07-01
1	2	13000	2014-07-03
2	5	8000	2014-07-03
3	2	13000	2014-07-04
4	4	35000	2014-07-05
1	3	22000	2014-07-07
4	3	22000	2014-07-07

- 위의 예시에서는 고객 번호만으로는 각 데이터들을 분리할 수 없다.
  - ex) ***고객번호 1번***인 데이터가 2개가 있기 때문이다. ***고객번호 1번***님~ 하고 부르면 ***도서번호 1***, ***도서번호 2***가 네! 하고 동시에 손들기 때문임
- 위의 경우, (***고객번호 + 도서번호***) 두 개의 속성 값을 조합하면 각 데이터들을 중복 없이 구분할 수 있다.
  - ex) ***고객번호 1번***이면서 ***도서번호 1번***이신 분~ 하면 딱 한 명밖에 손 안듦. → 성공!!
- 이러한 중복을 없애기 위한 것이 키 입니다

키의 종류

- **슈퍼키** : 각 데이터들을 식별할 수 있는 키들의 집합(여러 개 조합도 식별만 가능하면 삽가능)

고객

고객번호	이름	주민번호	주소	핸드폰
1	박지성	810101-1111111	영국 맨체스타	000-5000-0001
2	김연아	900101-2222222	대한민국 서울	000-6000-0001
3	장미란	830101-2333333	대한민국 강원도	000-7000-0001
4	추신수	820101-1444444	미국 클리블랜드	000-8000-0001

  - 예시) ***고객번호***만으로도 각 데이터들 중복 없이 식별 가능 / ***이름***으로도 가능 / ***주민번호***로도 가능 / ***고객번호 + 이름*** 으로도 가능 / ***주민번호 + 주소***로도 식별 가능 / 말 그대로 식별만 되면 어떻게 조합해도 상관 x

- **후보키:** 슈퍼키가 식별만 가능하면 몇 개씩 섞어서 사용해도 됐다면, 후보키는 그 중에서도 개수가 최소인 애들을 의미함.

고객

고객번호	이름	주민번호	주소	핸드폰
1	박지성	810101-1111111	영국 맨체스타	000-5000-0001
2	김연아	900101-2222222	대한민국 서울	000-6000-0001
3	장미란	830101-2333333	대한민국 강원도	000-7000-0001
4	추신수	820101-1444444	미국 클리블랜드	000-8000-0001

- 예시) ***고객 번호*** 하나만 있으면 다 식별 됨. → 개수 1개로 최소니까 고객번호는 ***후보키***
- 이름 하나만 있어도 마찬가지로 식별이 가능 → 개수가 1개로 최소니까 ***이름*** 또한 후보키

주문

고객번호	도서번호	판매가격	주문일자
1	1	7000	2014-07-01
1	2	13000	2014-07-03
2	5	8000	2014-07-03
3	2	13000	2014-07-04
4	4	35000	2014-07-05
1	3	22000	2014-07-07
4	3	22000	2014-07-07

- 예시) 위의 경우, (***고객번호 + 도서번호***)의 조합이 제일 개수가 최소이기 때문에 얘네가 후보키 중 하나로 들어가게 됨. 위의 테이블에서는 (***고객번호 + 판매 가격***) 또한 최소 개수로 각 데이터들을 식별할 수 있기 때문에 얘네도 후보키 중 하나임.

  💡위의 케이스와 같이 2개 이상의 속성을 이용한 키를 ***복합키***라고 함

- **기본키 :** 위에서 본 후보키들 중에서 딱 하나 선정해서 대표 시키는 키

고객

고객번호	이름	주민번호	주소	핸드폰
1	박지성	810101-1111111	영국 맨체스타	000-5000-0001
2	김연아	900101-2222222	대한민국 서울	000-6000-0001
3	장미란	830101-2333333	대한민국 강원도	000-7000-0001
4	추신수	820101-1444444	미국 클리블랜드	000-8000-0001

- 예시) ***고객번호***, ***이름***, ***주민번호***, ***주소***, ***핸드폰*** 각각이 후보키로 가능하지만, 난 ***고객번호***가 제일 맘에드니 얘를 ***기본키***로 할래!

[기본키 제약조건]

    - 각 데이터를 식별할 수 있는 고유값을 가져야 함
    - Null은 가질 수 없음
    - 키 값의 변동은 없어야 함.

- **대체키** : ***후보키***들 중에서 기본키가 되지 못한 키들(a.k.a 떨거지)
- **외래키** : 다른 테이블의 ***기본키***를 참조하는 속성

식별 관계 vs 비식별 관계

- 식별 관계(Identifying Relationship)
    - 다른 테이블의 ***기본키***를 참조해서 가져올 경우, 가져온 키로 데이터를 식별할 수 있는 경우를 의미함
- 비식별 관계(Non-Identifying Relationship)
    - 다른 테이블의 ***기본키***를 참조해서 가져오기는 하지만, 이걸로 데이터를 직접 식별하지는 않음.

관계

- 관계의 종류에는 ***1 : 1(일대일)***, ***1 : N(일대다)***, ***N : M (다대다)*** 관계가 존재한다.
- 그렇다면 오늘의 논제인 ***다대다관계***는 무엇이 문제일까?

다대다 관계(N : M)

[학생 테이블]

학번	이름	학과
1	홍길동	컴공과
2	장길산	토목과
3	임꺽정	불문과

학생 테이블

[과목 테이블]

과목코드	과목명	담당교수
S1	Java	조용준
S2	알고리즘	이몽룡
S3	Web	성춘향

과목 테이블

- 위의 케이스에서 [**학생 테이블**] 의 기본키는 ***학번***, [**과목 테이블**]의 기본키는 ***과목코드***이다.

- 한 학생은 여러 개의 과목을 들을 수 있고, 한 과목은 여러 학생에게 배정될 수 있다는 점에서 둘의 관계는 N : M 이다.
- [**학생 테이블**]에서 과목 테이블에 대한 정보까지 알기 위해선 [**과목 테이블**]을 Join해주어야 하며, [**과목 테이블**]에서 학생에 대한 정보를 알기 위해서도 마찬가지로 [**학생 테이블**]을 join해주어야 한다.

[학생 테이블]

학번	이름	학과	과목코드
1	홍길동	컴공과	S1
1	홍길동	컴공과	S2
2	장길산	토목과	S2
2	장길산	토목과	S3
3	임꺽정	불문과	

[과목 테이블]

과목코드	과목명	담당교수	수강학생
S1	Java	조용준	1
S2	알고리즘	이몽룡	1
S2	알고리즘	이몽룡	2
S3	Web	성춘향	3

- 위 그림은 [**학생 테이블**]에 [**과목 테이블**]을 join한 결과, 오른쪽은 [**과목 테이블**]에 [**학생 테이블**]을 join한 결과이다.
- 왼쪽 그림에서 기본키였던 ***학번***은 더는 기본키의 기능을 하지 못한다.(데이터 식별 불가)
- 오른쪽 그림에서 역시 ***과목 코드***도 기본 키의 기능을 하지 못한다.
- 위처럼 기본키가 제 역할을 못하고, 데이터 또한 중복되는 값이 늘어나기 때문에 N : M(다대다)관계는 해소해주어야 한다.

<< 그럼 해결은 어떻게 하는가? >>

- N : M 관계를 1 : N, M : 1 관계로 풀어주면 된다. 쉽게 말해 연관 테이블을 둘 사이에 넣어주면 되는 셈.

[학생 테이블] 1:N [수강 테이블] M:1 [과목 테이블]

학번	이름	학과
1	홍길동	컴공과
2	장길산	토목과
3	임꺽정	불문과

수강번호	학번	과목코드
1	1	S1
2	1	S2
3	2	S2
4	2	S3

과목코드	과목명	담당교수
S1	Java	조용준
S2	알고리즘	이몽룡
S3	Web	성춘향

💡 **그렇다면 우리 프로젝트에서 결제 주문에 대해 생각해보자.**

프로젝트 결제 / 주문 ERD 관련

- 편의상 **회원1**, **회원2**, **영양제1**, **영양제2**로 명명하겠다.

1. **회원과 주문은 1 : N 관계이다!**
    - 한 회원은 여러 개의 주문을 만들 수 있다.
    - 회원1은 오늘 주문한 후, 다음주 토요일에 또 주문할 수 있기 때문이다.
2. 주문과 상품은 N : M 관계이다!
    - 주문 하나에는 여러 개의 상품이 담긴다.
        - ex) 회원1은 오늘 (**영양제1 + 영양제2 + 영양제3**) 이렇게 총 3개의 영양제를 주문했다.
    - 한 영양제는 여러 개의 주문에 담길 수 있다.
        - ex) 영양제1은 회원1, 회원4, 회원6이 주문했다.

    → 그렇다면? 당연히 이 관계를 해소해줘야지!!!!

    주문과 주문 옵션(1:N)

    - 한 주문에는 여러 개의 주문 옵션이 연결될 수 있다.
        - ***회원1***의 ***주문1***에는 (***주문1 + 영양제1***), (***주문1 + 영양제2***)와 같은 상세 내역(***주문 옵션***)이 연결된다.
    - 반대로 주문 옵션 하나에는 여러 개의 주문이 연결될 수 없다.

    영양제와 주문 옵션(1 : N)

    - 한 영양제는 여러 개의 주문 옵션과 연결될 수 있다.
        - ***영양제1***은 주문 옵션의 (***주문1 + 영양제1***), (***주문2 + 영양제1***) 과 같이 여러 개의 주문과 연결될 수 있다.
    - 반대로 한 개의 주문 옵션에는 여러 개의 영양제가 연결될 수 없다.

회원 - 장바구니 - 영양제

- 한 회원은 장바구니 하나만 가질 수 있다.(반대도 마찬가지)
- 한 장바구니에는 여러 개의 영양제가 담길 수 있다.
- 한 영양제는 여러 개의 장바구니에 담길 수 있다.

>> 이 경우 ***회원-장바구니(1 : 1)***, ***장바구니-영양제(N : M)*** 의 관계를 생각할 수 있고, 그렇다면 ***장바구니-영양제***의 다대다 관계를 해소주는게 맞지 않겠는가?